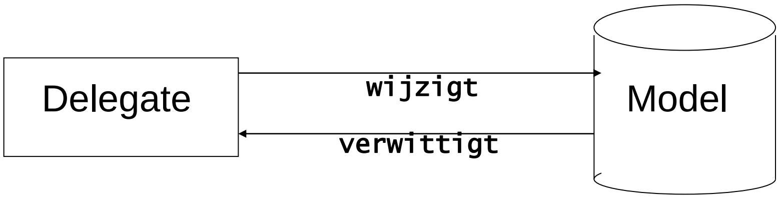
HoGent
BEDRIJF
EN
ORGANISATIE

MODEL-VIEW-CONTROLLER
DEEL 3
TableView

1. Delegate-Model architectuur

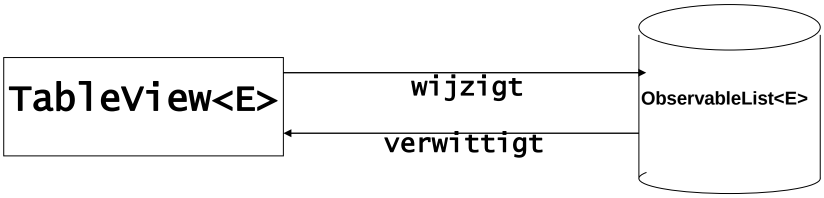


Delegate: View en controller worden samengebracht tot één enkel object, de delegate.

Model: de data

Het **delegate-model** wordt in javaFx-componenten toegepast, zoals ListView, **TableView** en TreeView.

2. TableView<E>



- **TableView<E>** voorziet alle functionaliteiten om data in tabelvorm te presenteren en die data interactief te wijzigen.

- **TableView<E>** ondersteunt de scheiding van data, presentatie en invoerverwerking, nl. de **delegate-model** archictectuur.

HoGent

2. TableView<E>

- Via de **TableView<E> GUI component** worden wijzigingen van de data doorgeven aan de **observableList**.

- De **observableList** verwittigt (notifies) de tableView-component bij wijziging, toevoeging en verwijdering van de data.

- De interface **ObservableList<E>** (het model) voorziet methoden voor het opvragen van de data.

- De **tableView** (de delegate) ondervraagt het model voor de opbouw van de tabel en zal het model wijzigen op basis van user input.

HoGent

3. TableView<E> & ObservableList<E>

- Interface ObservableList<E>
  extends java.util.List<E>, Observable

3. TableView<E> & ObservableList<E>

- ObservableList<E>
  - Het type van de attributen van klasse E zijn ***package javafx.beans.property***
    - **SimpleBooleanProperty**
    - **SimpleDoubleProperty**
    - **SimpleFloatProperty**
    - **SimpleIntegerProperty**
    - **SimpleLongProperty** en/of
    - **SimpleStringProperty**

- SimpleBooleanProperty, SimpleDoubleProperty, SimpleFloatProperty, SimpleIntegerProperty, SimpleLongProperty, SimpleStringProperty

Dient voor de koppeling (binding) tussen de 'data' en de 'cellen van een kolom in de tableView'

***Bv.* in kolom 'firstNameCol' worden alle voornamen van Person weergegeven.**

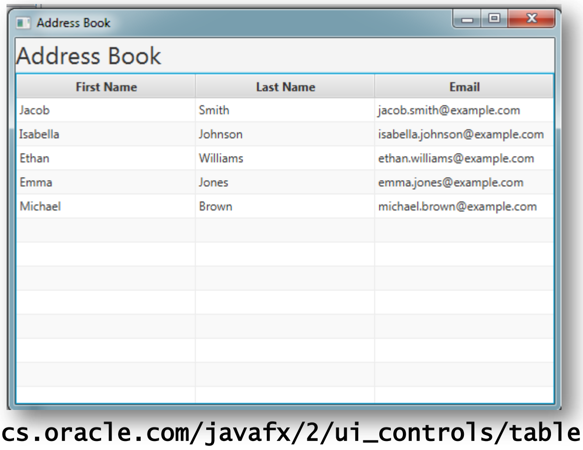
- **addresBookTable is een TableView<Person>**
- **We koppelen de TableView met een ObservableList<Person>**
  addressBookTable.setItems(*een ObservableList<Person>*);
- **Klasse Person bevat het attribuut firstName. 'firstName' is van het type SimpleStringProperty.**
- **De kolom firstNameCol van de tableView wordt gekoppeld met de property firstName.**

```
firstNameCol.setCellValueFactory(cellData ->
        cellData.getValue().firstNameProperty());
```

Jacob
Isabella
Ethan
Emma
Michael

4. TableView<E> & ObservableList<E>

- Voorbeeld: een tabel van personen wordt weergegeven



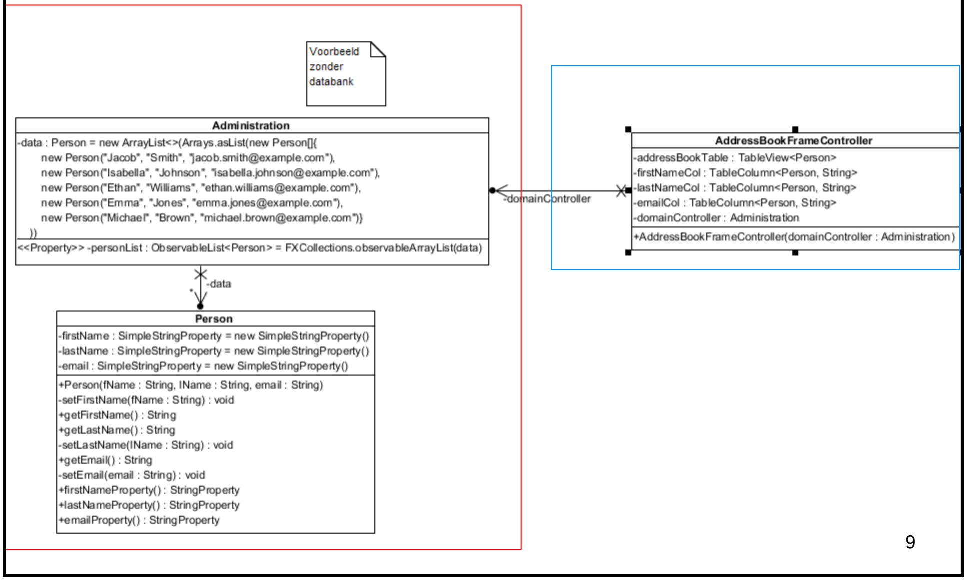
Address Book

First Name	Last Name	Email
Jacob	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example.com
Ethan	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com

http://docs.oracle.com/javafx/2/ui_controls/table-view.htm

package domein

package gui

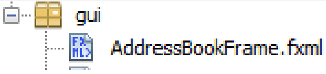


```
Administration
-data : Person = new ArrayList<>(Arrays.asList(new Person[]{
      new Person("Jacob", "Smith", "jacob.smith@example.com"),
      new Person("Isabella", "Johnson", "isabella.johnson@example.com"),
      new Person("Ethan", "Williams", "ethan.williams@example.com"),
      new Person("Emma", "Jones", "emma.jones@example.com"),
      new Person("Michael", "Brown", "michael.brown@example.com")}
   ))
<<Property>> -personList : ObservableList<Person> = FXCollections.observableArrayList(data)
```

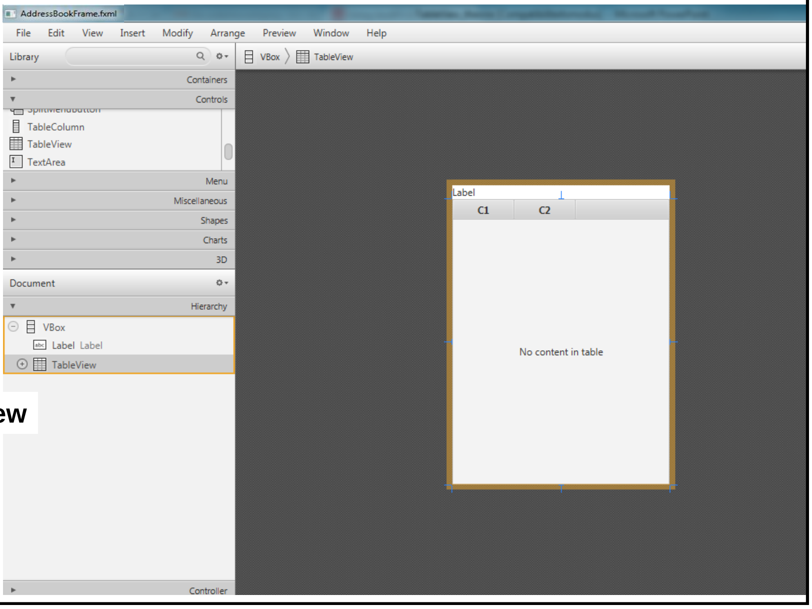
```
Person
-firstName : SimpleStringProperty = new SimpleStringProperty()
-lastName : SimpleStringProperty = new SimpleStringProperty()
-email : SimpleStringProperty = new SimpleStringProperty()
+Person(fName : String, lName : String, email : String)
-setFirstName(fName : String) : void
+getFirstName() : String
+getLastName() : String
-setLastName(lName : String) : void
+getEmail() : String
-setEmail(email : String) : void
+firstNameProperty() : StringProperty
+lastNameProperty() : StringProperty
+emailProperty() : StringProperty
```

```
AddressBookFrameController
-addressBookTable : TableView<Person>
-firstNameCol : TableColumn<Person, String>
-lastNameCol : TableColumn<Person, String>
-emailCol : TableColumn<Person, String>
-domainController : Administration
+AddressBookFrameController(domainController : Administration)
```

9



Scene Builder



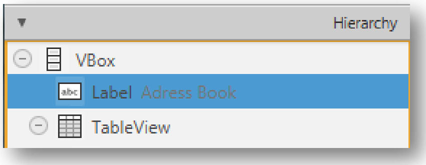
VBox

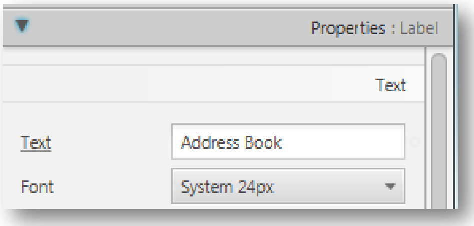
Label

TableView

Scene Builder

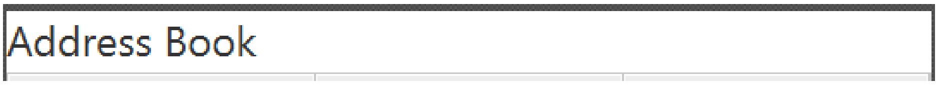
Label:





Text: Address Book

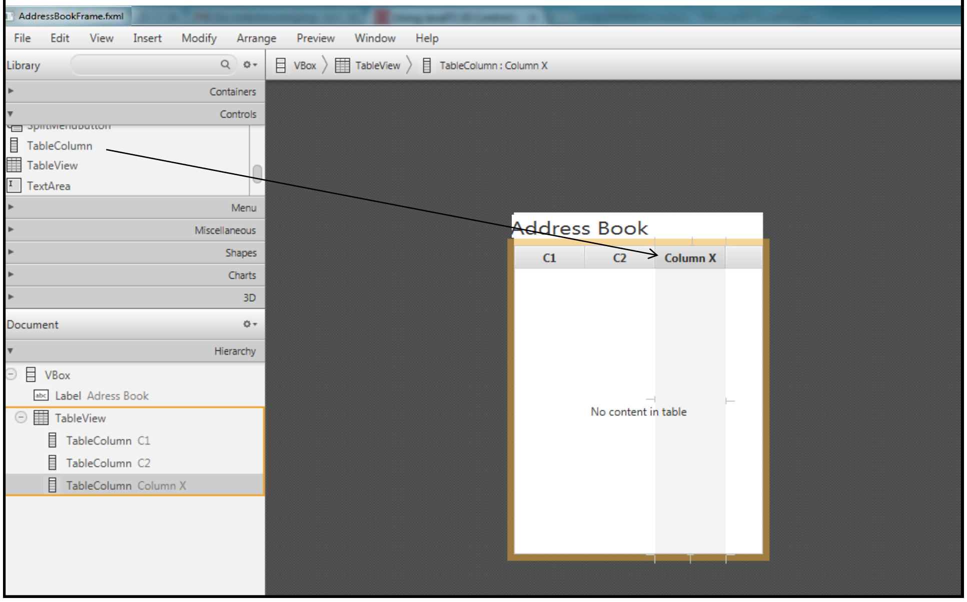
Font: System 24px



HoGent

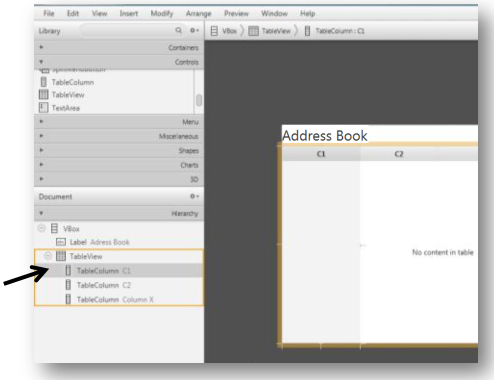
Scene Builder

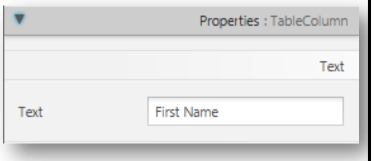
TableView:



Scene Builder

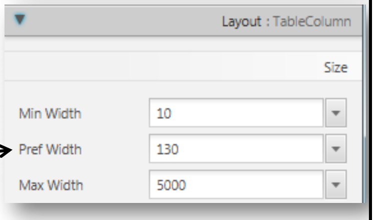
Eerste TableColumn:





Text

Text: First Name

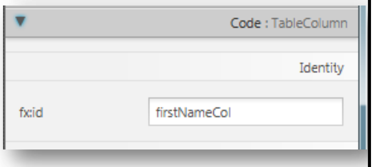


Size

Min Width: 10

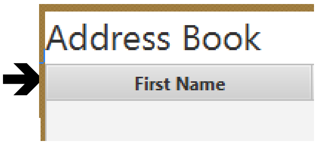
Pref Width: 130

Max Width: 5000



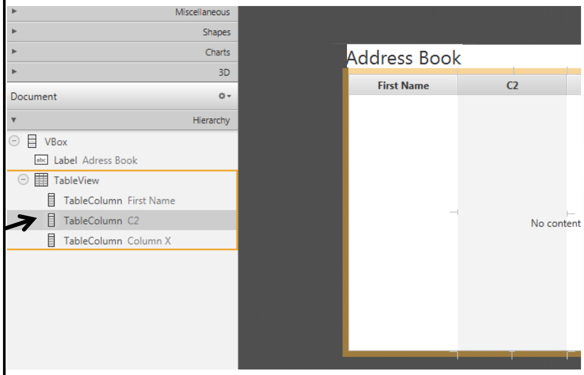
Identity

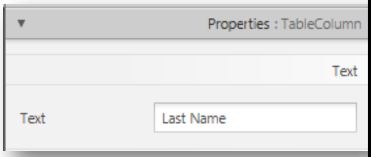
fx:id: firstNameCol



Scene Builder

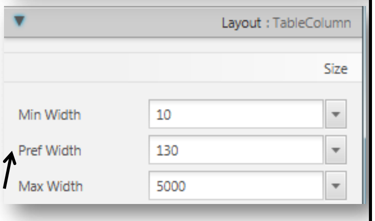
Tweede TableColumn:





Text

Text: Last Name

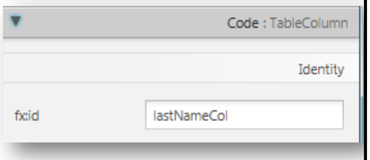


Size

Min Width: 10

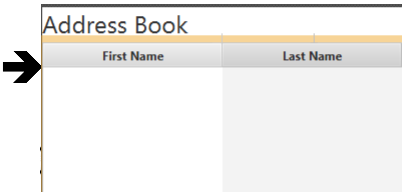
Pref Width: 130

Max Width: 5000



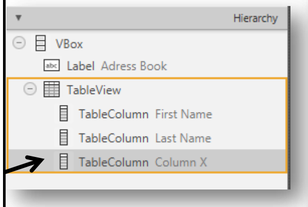
Identity

fx:id: lastNameCol

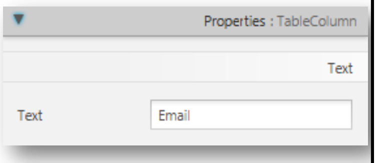


Scene Builder

Derde TableColumn:

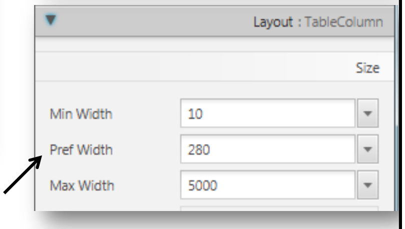

Hierarchy

- VBox
  - Label Adress Book
  - TableView
    - TableColumn First Name
    - TableColumn Last Name
    - TableColumn Column X


Properties : TableColumn

Text

Text: Email

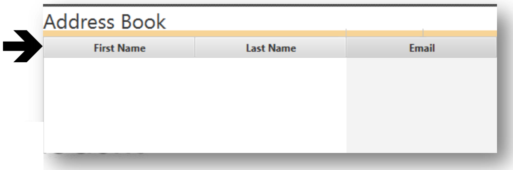

Layout : TableColumn

Size

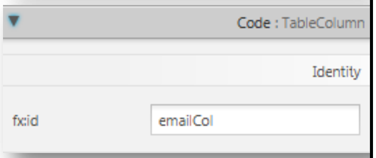
Min Width: 10

Pref Width: 280

Max Width: 5000


Address Book

First Name	Last Name	Email

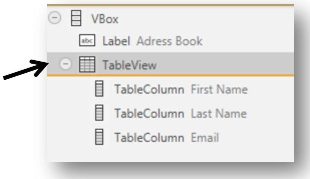

Code : TableColumn

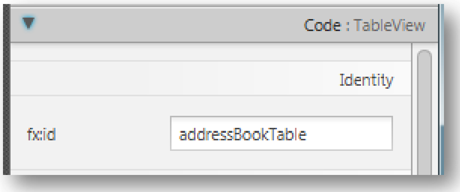
Identity

fx:id: emailCol

Scene Builder

TableView:


- VBox
  - Label Adress Book
  - TableView
    - TableColumn First Name
    - TableColumn Last Name
    - TableColumn Email

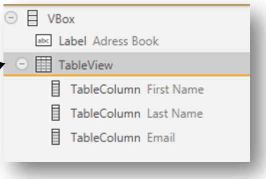

Code : TableView

Identity

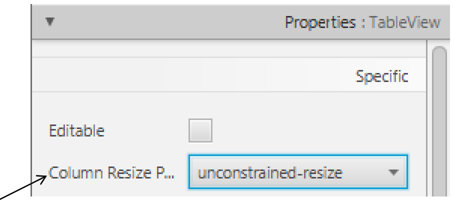
fx:id: addressBookTable

Scene Builder

TableView:



- VBox
  - Label Adress Book
  - TableView
    - TableColumn First Name
    - TableColumn Last Name
    - TableColumn Email



Properties : TableView

Specific

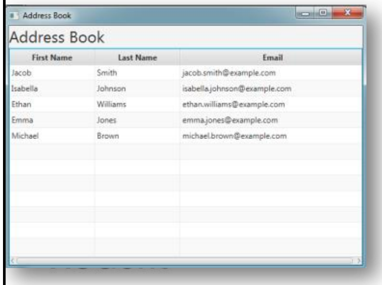
Editable ☐

Column Resize P... unconstrained-resize

default

➔ Run:

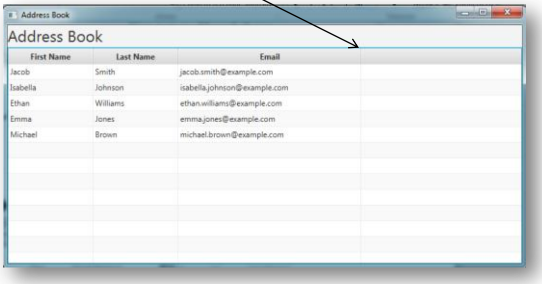
resize (vierde kolom):



Address Book

Address Book

First Name	Last Name	Email
Jacob	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example.com
Ethan	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com



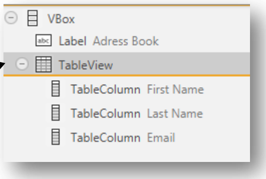
Address Book

Address Book

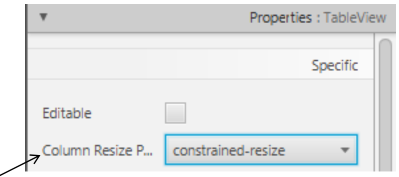
First Name	Last Name	Email
Jacob	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example.com
Ethan	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com

Scene Builder

TableView:



- VBox
  - Label Adress Book
  - TableView
    - TableColumn First Name
    - TableColumn Last Name
    - TableColumn Email



Properties : TableView

Specific

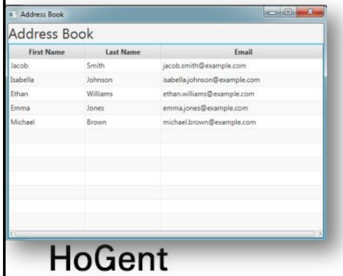
Editable ☐

Column Resize P... constrained-resize

Wijzigen naar ‘constrained-resize’

➔ Run:

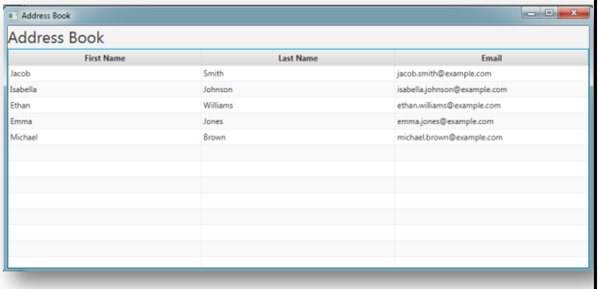
resize (3 kolommen worden groter):



Address Book

Address Book

First Name	Last Name	Email
Jacob	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example.com
Ethan	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com



Address Book

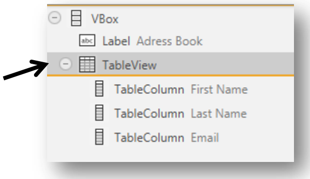
Address Book

First Name	Last Name	Email
Jacob	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example.com
Ethan	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com

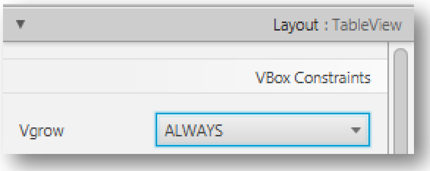
HoGent

Scene Builder

TableView:



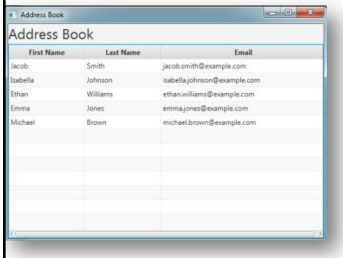
- VBox
  - Label Adress Book
  - TableView
    - TableColumn First Name
    - TableColumn Last Name
    - TableColumn Email



Layout : TableView

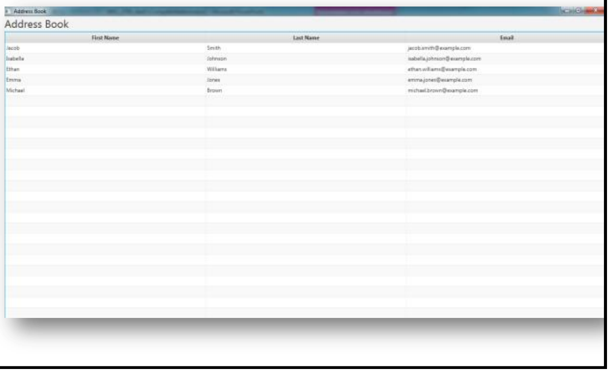
VBox Constraints

Vgrow: ALWAYS



Address Book

First Name	Last Name	Email
Jacob	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example.com
Ethan	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com

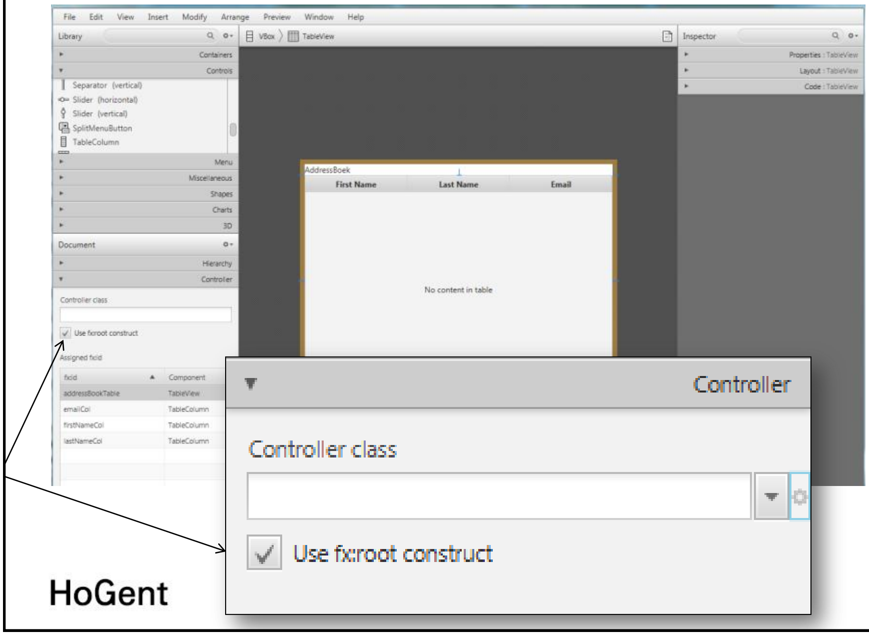


Address Book

First Name	Last Name	Email
Jacob	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example.com
Ethan	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com

HoGent

Scene Builder



AddressBoek

First Name	Last Name	Email

No content in table

Controller class

☑ Use fx:root construct

Assigned fx:id

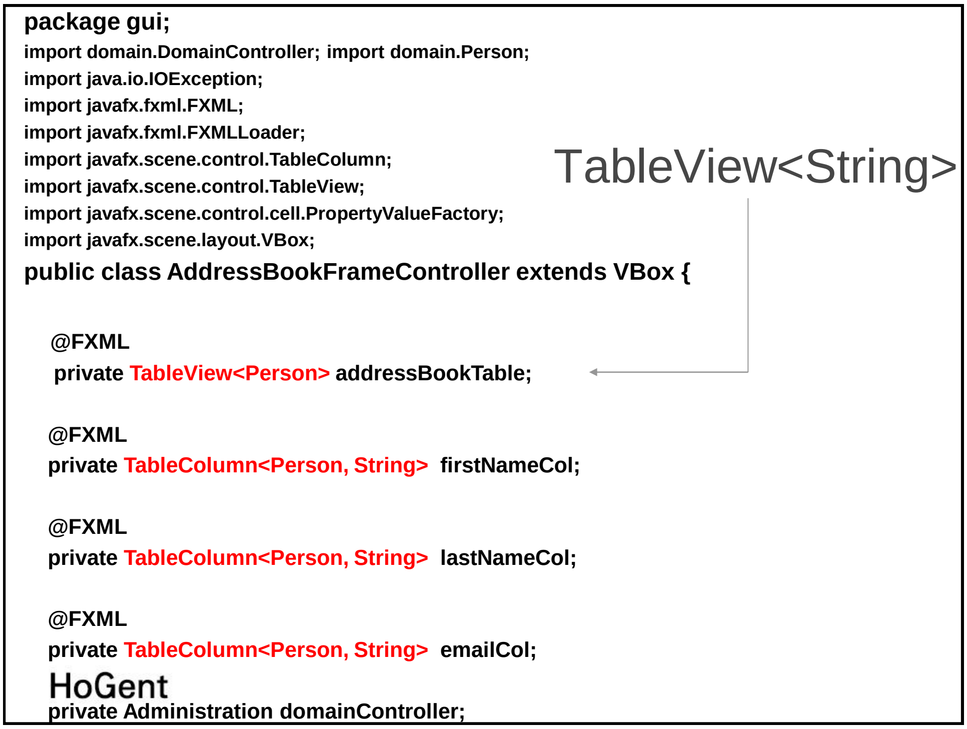
fx:id	Component
addressBookTable	TableView
emailCol	TableColumn
firstNameCol	TableColumn
lastNameCol	TableColumn

Controller

Controller class

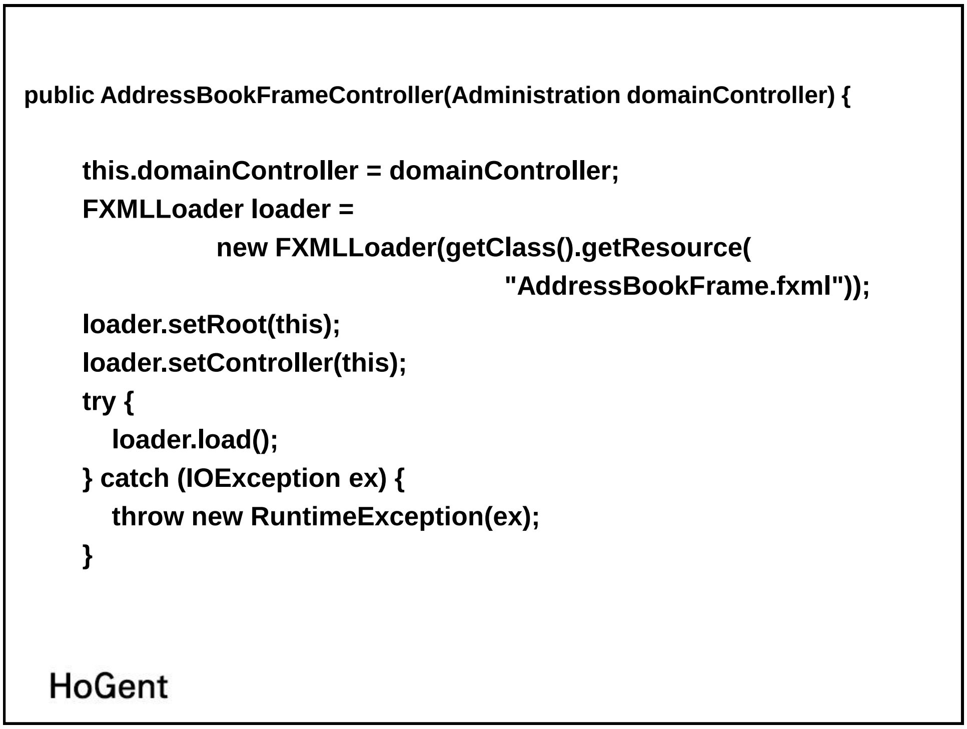
☑ Use fx:root construct

HoGent



```
package gui;
import domain.DomainController; import domain.Person;
import java.io.IOException;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.VBox;
public class AddressBookFrameController extends VBox {

    @FXML
     private TableView<Person> addressBookTable;

    @FXML
    private TableColumn<Person, String>  firstNameCol;

    @FXML
    private TableColumn<Person, String>  lastNameCol;

    @FXML
    private TableColumn<Person, String>  emailCol;

    private Administration domainController;
```

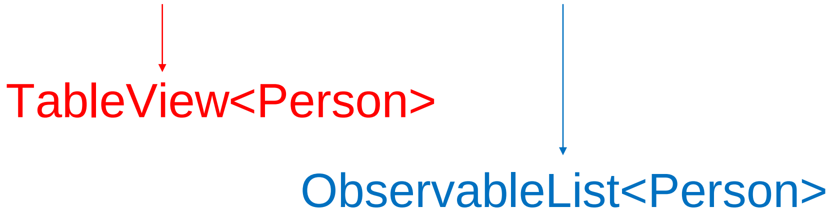
TableView<String>



```
public AddressBookFrameController(Administration domainController) {

        this.domainController = domainController;
        FXMLLoader loader =
                        new FXMLLoader(getClass().getResource(
                                                "AddressBookFrame.fxml"));
        loader.setRoot(this);
        loader.setController(this);
        try {
            loader.load();
        } catch (IOException ex) {
            throw new RuntimeException(ex);
        }
```

```
/*De eerste kolom verbinden met de property “firstName” van de klasse
Person. */
          firstNameCol.setCellValueFactory(cellData ->
                    cellData.getValue().firstNameProperty());

//Analoog tweede kolom:
      lastNameCol.setCellValueFactory(cellData ->
          cellData.getValue().lastNameProperty());

//Analoog derde kolom:
      emailCol.setCellValueFactory(cellData ->
                    cellData.getValue().emailProperty());
```

```
      //tableView opvullen met data
      addressBookTable.setItems(domainController.getPersonList());

   }
}
```



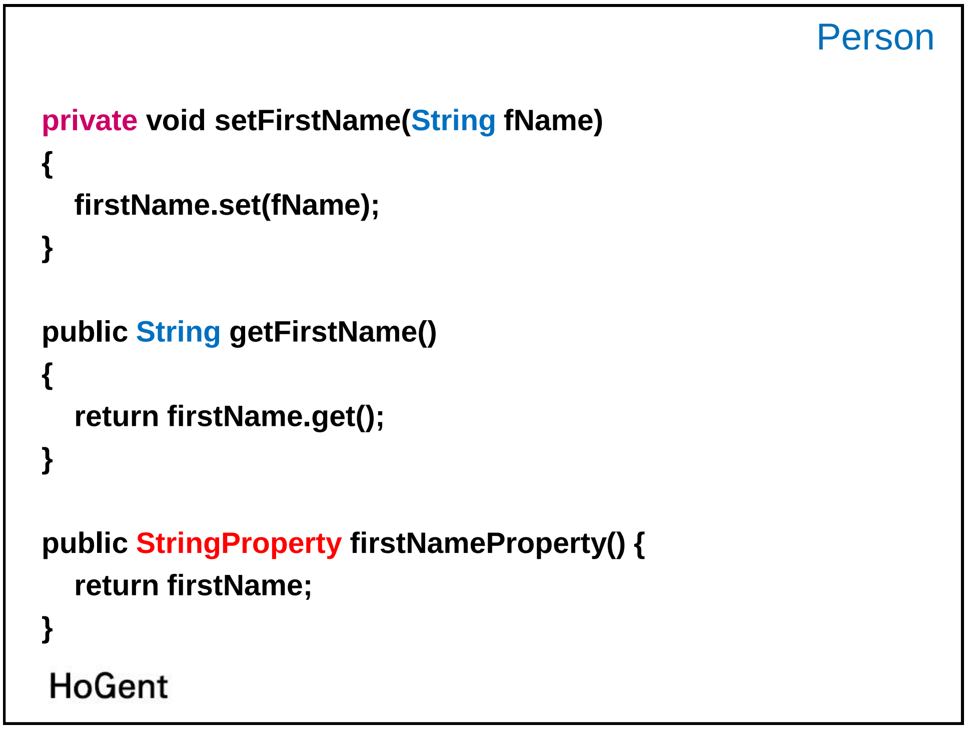
TableView<Person>

ObservableList<Person>

```
package domain;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
public class Administration {

   private ObservableList<Person> personList;
  //Voorbeeld zonder databank
   private static final List<Person> data = new ArrayList<>(Arrays.asList(new Person[]{
      new Person("Jacob", "Smith", "jacob.smith@example.com"),
      new Person("Isabella", "Johnson", "isabella.johnson@example.com"),
      new Person("Ethan", "Williams", "ethan.williams@example.com"),
      new Person("Emma", "Jones", "emma.jones@example.com"),
      new Person("Michael", "Brown", "michael.brown@example.com")}
   ));

   public Administration() {
      personList = FXCollections.observableArrayList(data);
   }
   public ObservableList<Person> getPersonList(){
      return FXCollections.unmodifiableObservableList(personList);
   }
}
```

Person

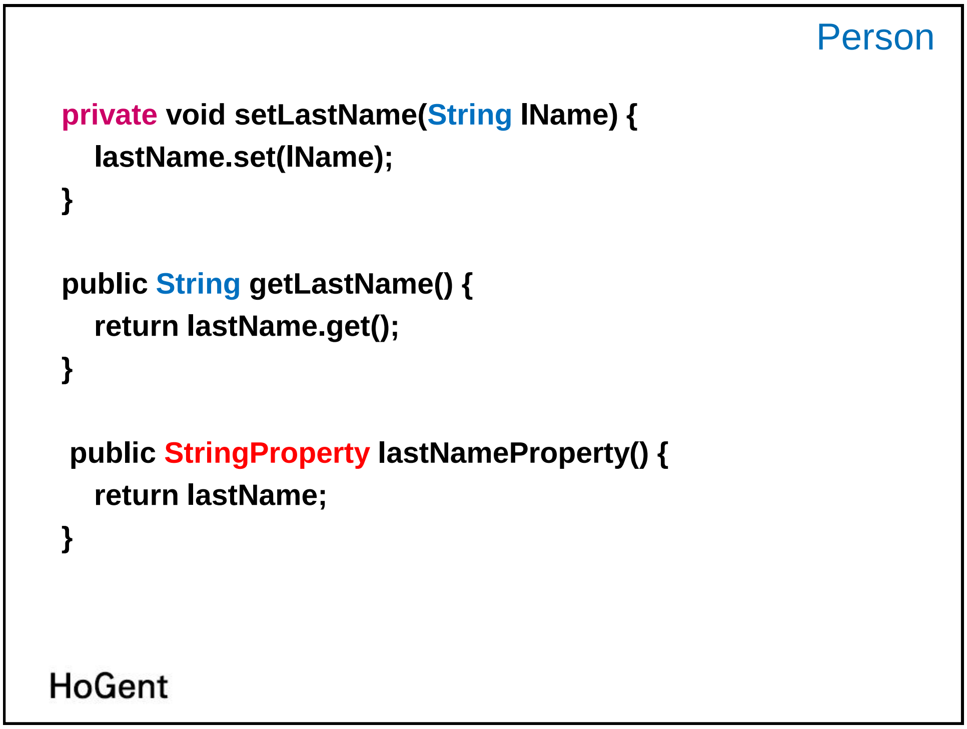
```
package domain;
import javafx.beans.property.SimpleStringProperty;

public class Person {

    private final SimpleStringProperty firstName =
                               new SimpleStringProperty();
    private final SimpleStringProperty lastName =
                               new SimpleStringProperty();
    private final SimpleStringProperty email =
                               new SimpleStringProperty();

    public Person(String fName, String lName, String email) {
        setFirstName(fName);
        setLastName(lName);
        setEmail(email);
    }
```



Person

```
private void setFirstName(String fName)
{
   firstName.set(fName);
}

public String getFirstName()
{
   return firstName.get();
}

public StringProperty firstNameProperty() {
   return firstName;
}
```

HoGent



Person

```
private void setLastName(String lName) {
   lastName.set(lName);
}

public String getLastName() {
   return lastName.get();
}

 public StringProperty lastNameProperty() {
   return lastName;
}
```

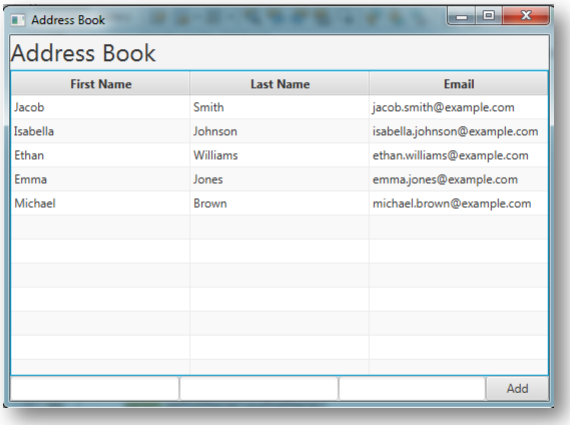
HoGent

Person

```
private void setEmail(String email) {
    this.email.set(email);
}

public String getEmail() {
    return email.get();
}

 public StringProperty emailProperty() {
    return email;
}
}
```

HoGent

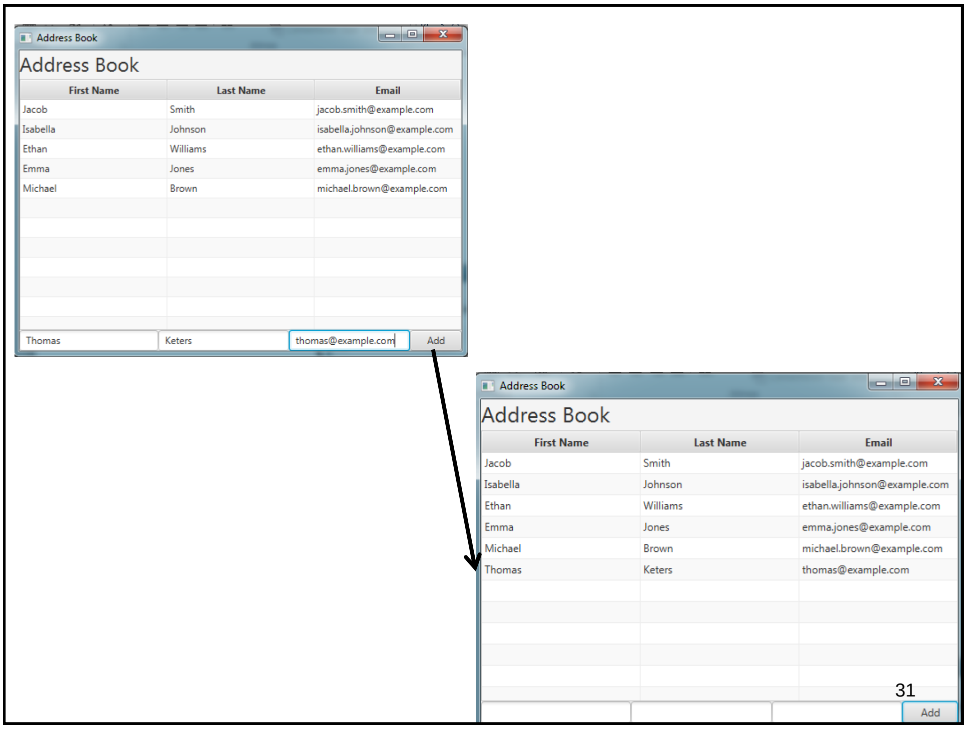
4.1 Een rij toevoegen (zonder validatie)



First Name	Last Name	Email
Jacob	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example.com
Ethan	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com

http://docs.oracle.com/javafx/2/ui_controls/table-view.htm

30



Address Book

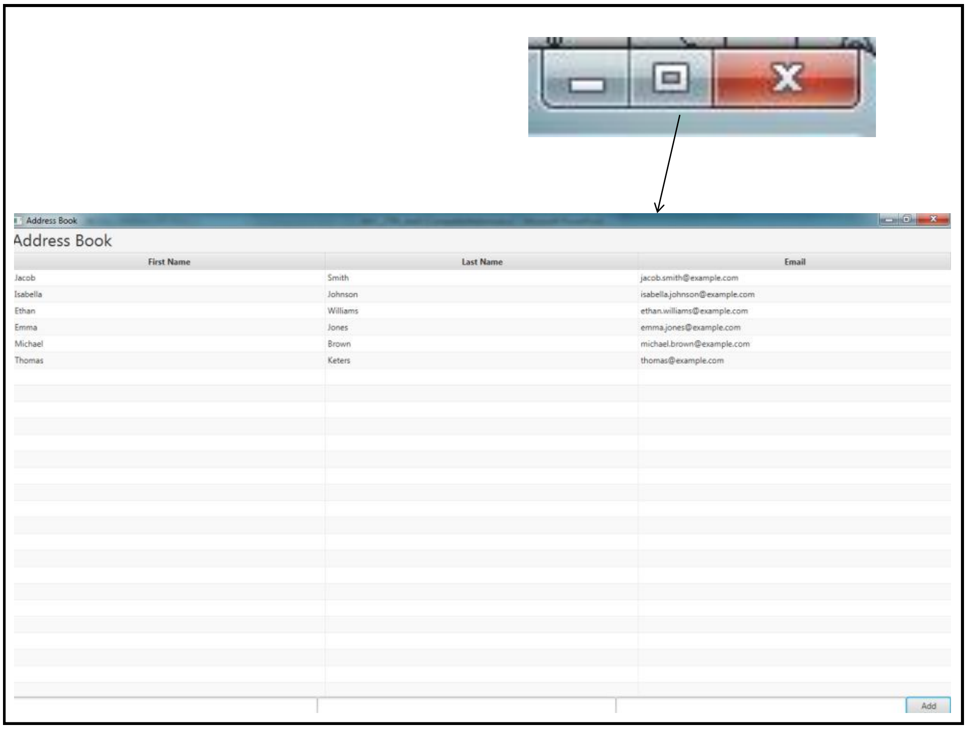
First Name	Last Name	Email
Jacob	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example.com
Ethan	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com

Thomas	Keters	thomas@example.com	Add

Address Book

First Name	Last Name	Email
Jacob	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example.com
Ethan	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com
Thomas	Keters	thomas@example.com

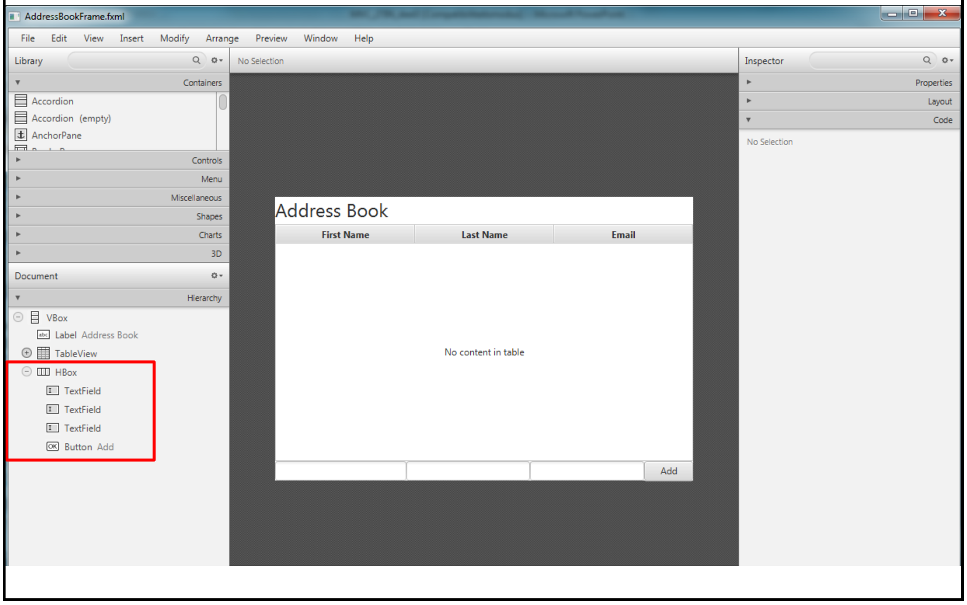
31



Address Book

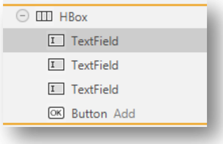
First Name	Last Name	Email
Jacob	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example.com
Ethan	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com
Thomas	Keters	thomas@example.com

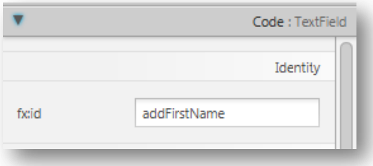
Scene Builder



Scene Builder

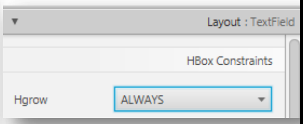
TextFields




Code : TextField

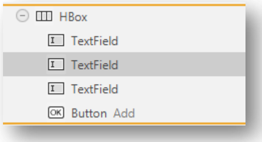
Identity

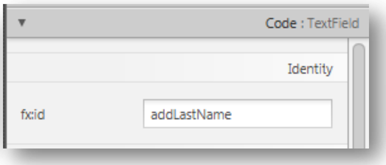
fx:id addFirstName


Layout : TextField

HBox Constraints

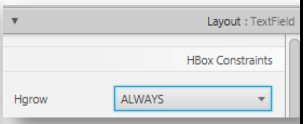
Hgrow ALWAYS




Code : TextField

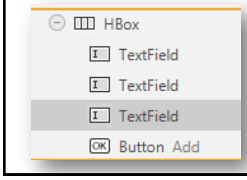
Identity

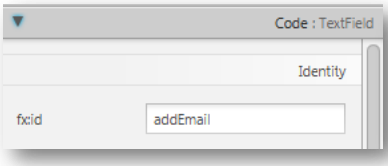
fx:id addLastName


Layout : TextField

HBox Constraints

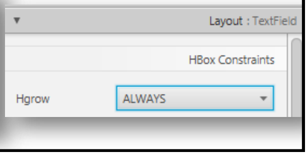
Hgrow ALWAYS




Code : TextField

Identity

fx:id addEmail

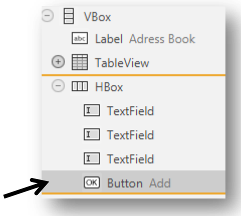

Layout : TextField

HBox Constraints

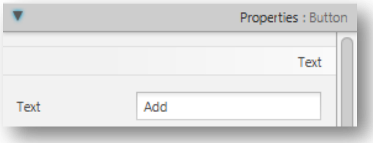
Hgrow ALWAYS

Scene Builder

Button



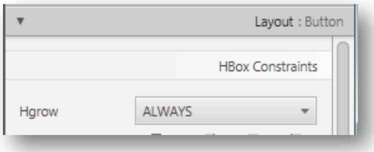
VBox
Label Adress Book
TableView
HBox
TextField
TextField
TextField
Button Add



Properties : Button

Text

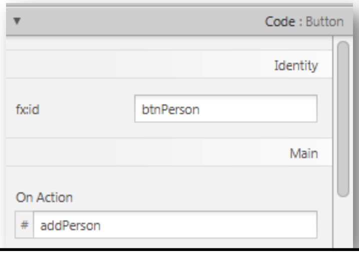
Text: Add



Layout : Button

HBox Constraints

Hgrow: ALWAYS



Code : Button

Identity

fx:id: btnPerson

Main

On Action: # addPerson

HoGent

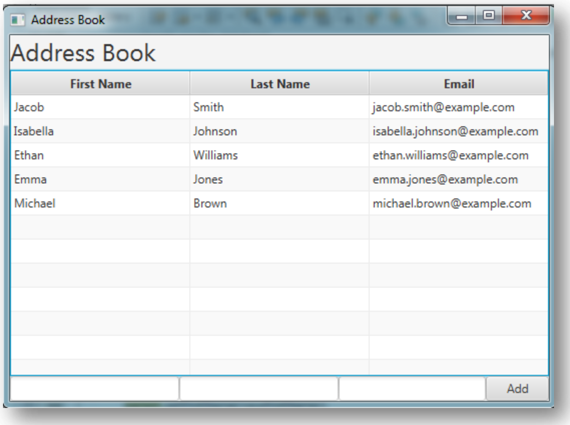
AddressBookFrameController

```
@FXML
private TextField addFirstName;

@FXML
private TextField addLastName;

@FXML
private TextField addEmail;

@FXML
private Button btnPerson;

@FXML
private void addPerson(ActionEvent event) {
    domainController.addPerson(addFirstName.getText(),
                                addLastName.getText(), addEmail.getText());
    addFirstName.clear();
    addLastName.clear();
    addEmail.clear();
}
```

Administration

```
public void addPerson(String firstName,
                      String lastName, String email) {
  personList.add(
      new Person(firstName, lastName, email));
}
```

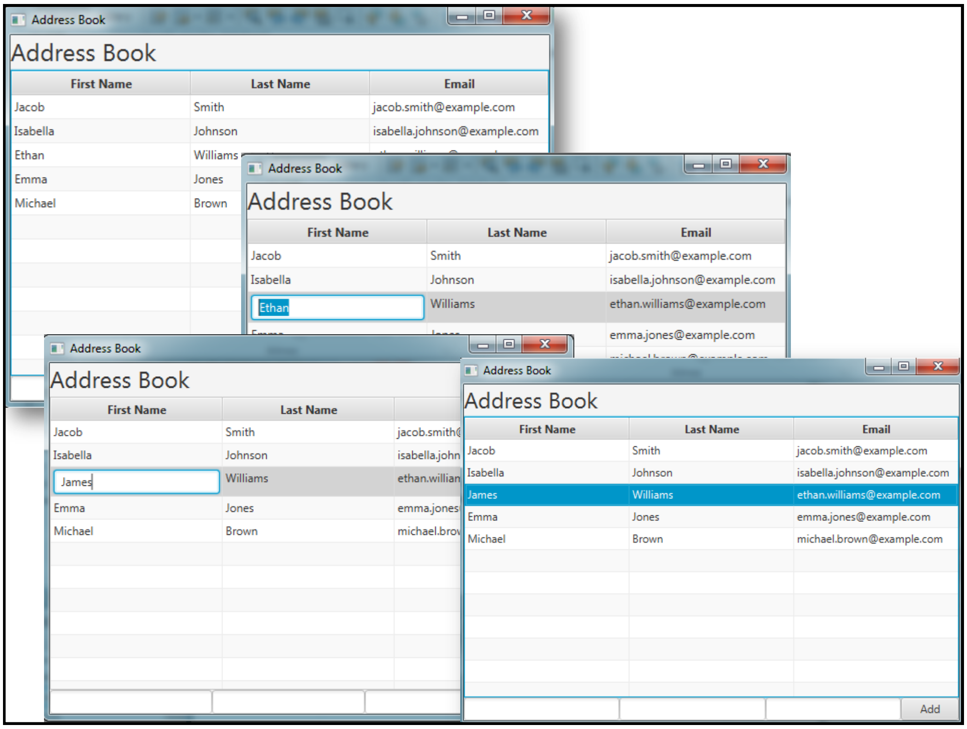
4.2 Een cel wijzigen (zonder validatie)



Address Book

First Name	Last Name	Email
Jacob	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example.com
Ethan	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com

http://docs.oracle.com/javafx/2/ui_controls/table-view.htm

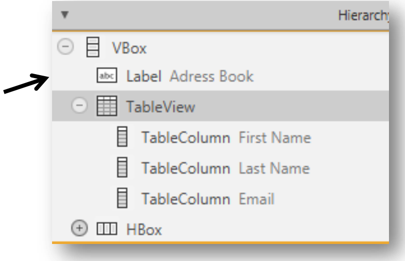


Address Book

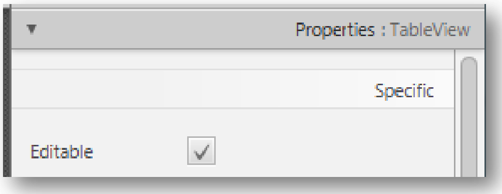
First Name	Last Name	Email
Jacob	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example.com
James	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com

Scene Builder

TableView



- VBox
  - Label Adress Book
  - TableView
    - TableColumn First Name
    - TableColumn Last Name
    - TableColumn Email
  - HBox

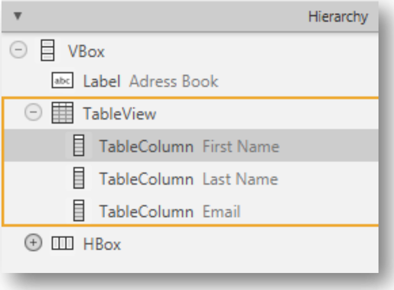


Editable ☑

HoGent

Scene Builder

TableColumn First Name



Hierarchy

VBox

Label Adress Book

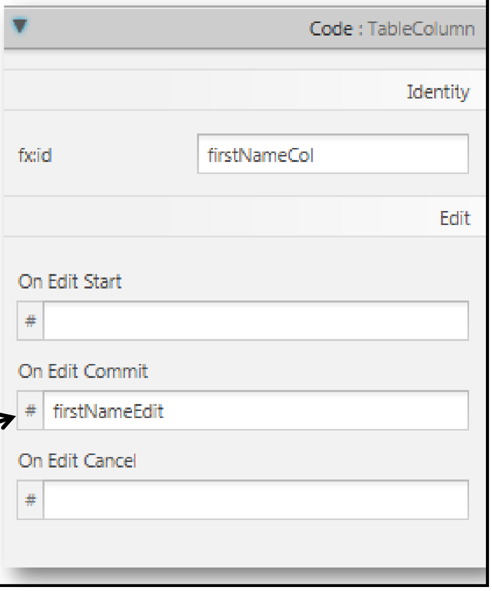
TableView

TableColumn First Name

TableColumn Last Name

TableColumn Email

HBox



Code : TableColumn

Identity

fx:id: firstNameCol

Edit

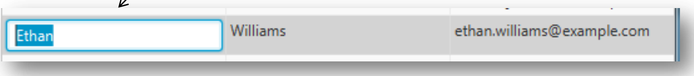
On Edit Start: #

On Edit Commit: # firstNameEdit

On Edit Cancel: #

AddressBookFrameController

```
public AddressBookFrameController(Administration domainController) {
    …
    firstNameCol.setCellValueFactory(cellData ->
                    cellData.getValue().firstNameProperty());

    firstNameCol.setCellFactory(TextFieldTableCell.forTableColumn());
    …
}
```



Ethan	Williams	ethan.williams@example.com

Bij dubbelklik in een cel van firstNameCol, verandert de cel in een textfield

AddressBookFrameController

```
@FXML
private void firstNameEdit(CellEditEvent<Person, String> event) {
    String newFirstName = event.getNewValue();
    int index = event.getTablePosition().getRow();
    domainController.editFirstName(index, newFirstName);
    addressBookTable.getSelectionModel().clearSelection();
}
```

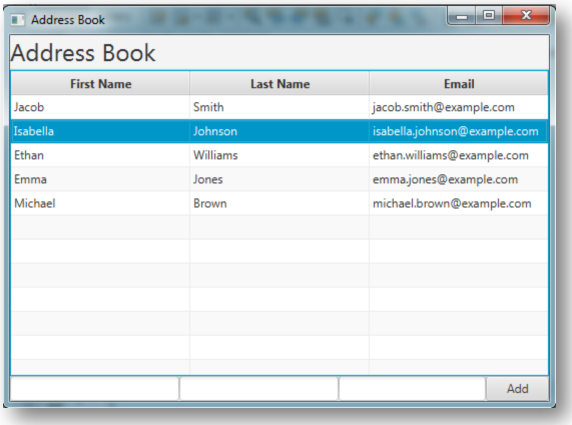
Administration

```
public void editFirstName(int index, String newFirstName)
{
    personList.get(index).setFirstName(newFirstName);
}
```

Person

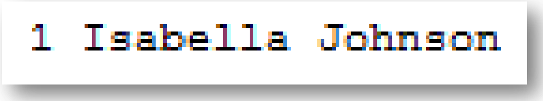
```
protected void setFirstName(String fName) {
        firstName.set(fName);   }
```

4.3 Een rij selecteren



Address Book

First Name	Last Name	Email
Jacob	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example.com
Ethan	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com



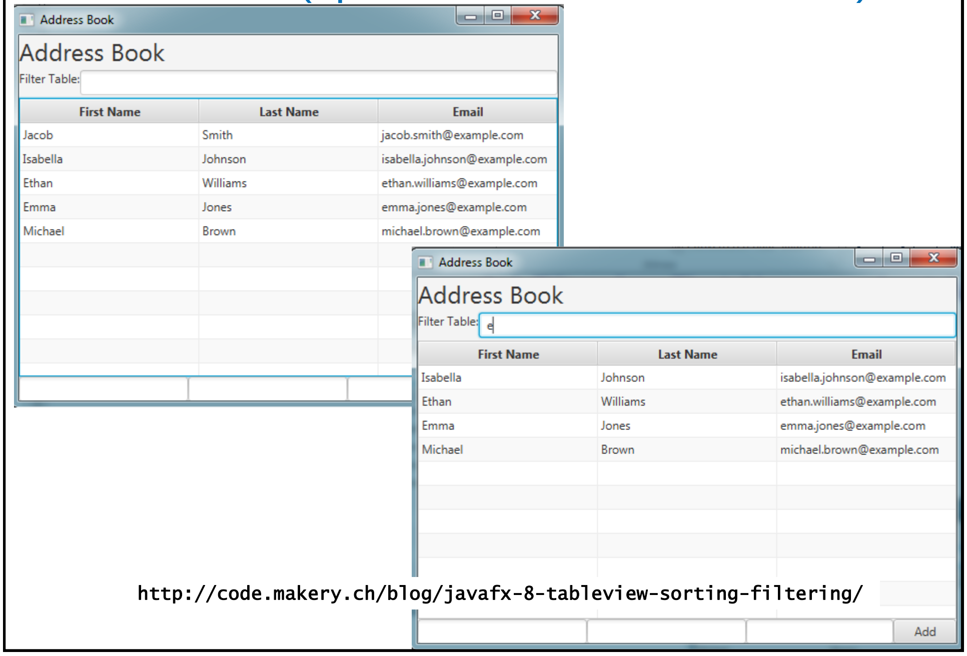
```
1 Isabella Johnson
```

AddressBookFrameController

```
public AddressBookFrameController(Administration domainController) {
    …
     //Een listener toevoegen
      addressBookTable.getSelectionModel().selectedItemProperty().
                addListener((observableValue, oldPerson, newPerson) -> {
                    //Controleer of er een persoon is geselecteerd
                    if (newPerson != null) {
                             int index = addressBookTable.
                                                    getSelectionModel().getSelectedIndex();
                           System.out.printf("%d %s %s\n  ", index,
                                  newPerson.getFirstName(), newPerson.getLastName());
                   }
      });
}
```

HoGent

4.4 Filter (op First Name en Last Name)



Address Book

Filter Table:

First Name	Last Name	Email
Jacob	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example.com
Ethan	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com

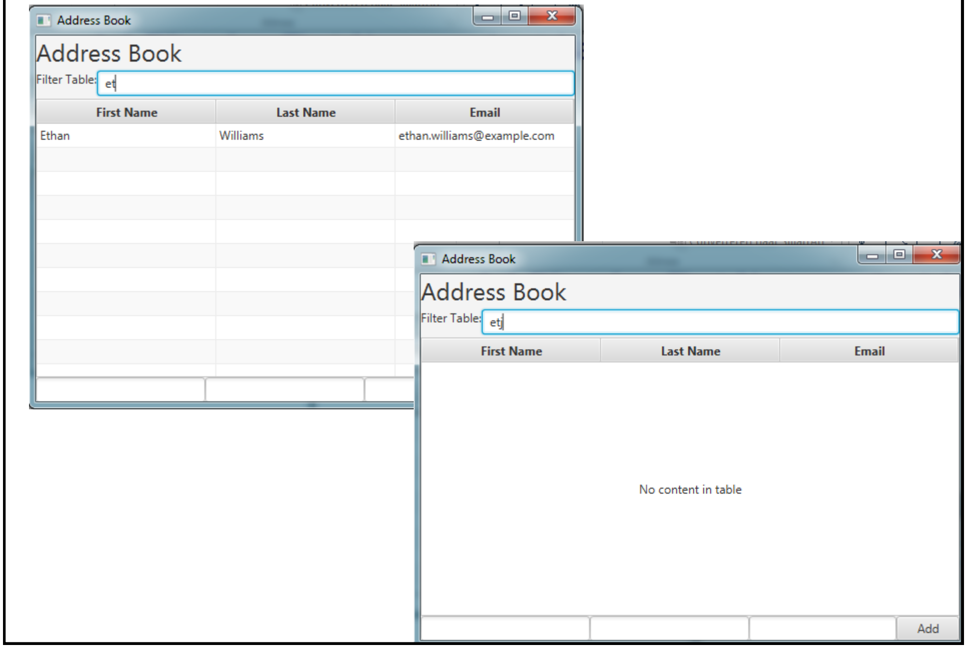
Address Book

Filter Table: e

First Name	Last Name	Email
Isabella	Johnson	isabella.johnson@example.com
Ethan	Williams	ethan.williams@example.com
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.com

http://code.makery.ch/blog/javafx-8-tableview-sorting-filtering/

4.4 Filter (op First Name en Last Name)



Filter Table: et

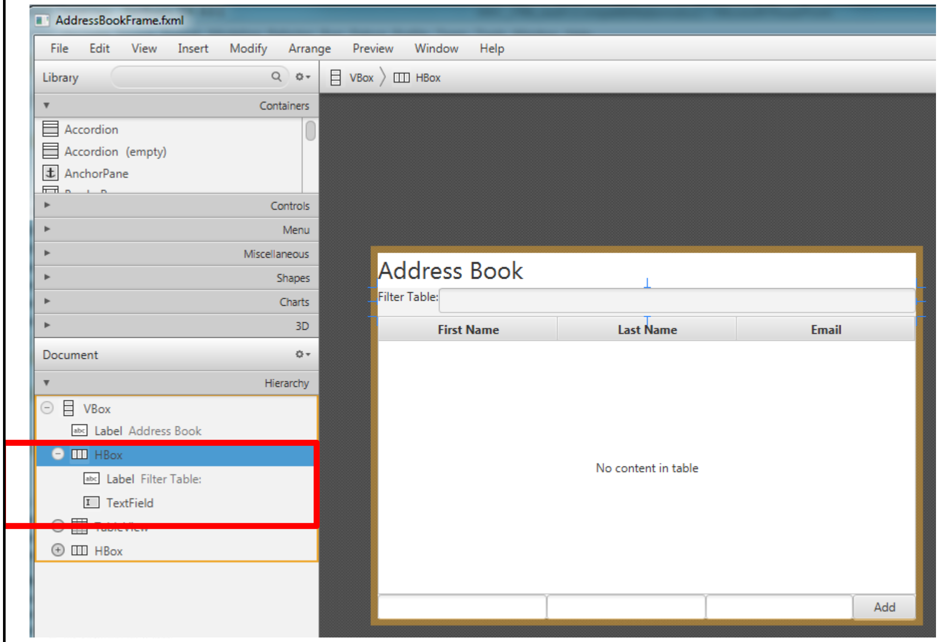
First Name	Last Name	Email
Ethan	Williams	ethan.williams@example.com

Filter Table: etj

First Name	Last Name	Email

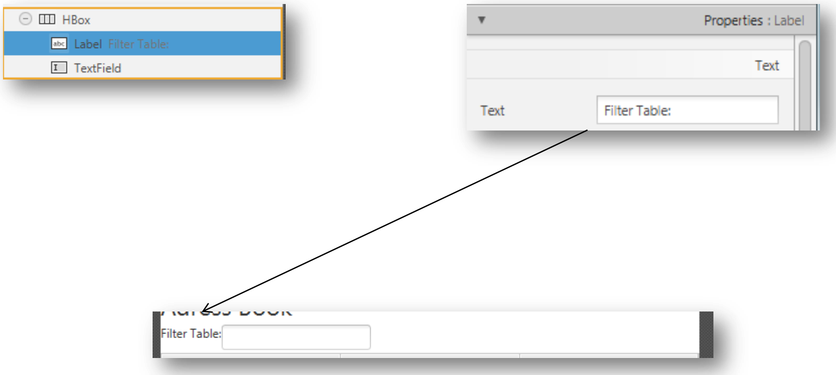
No content in table

Scene Builder



Scene Builder

Label



HBox
- Label Filter Table:
- TextField

Properties : Label

Text

Text: Filter Table:

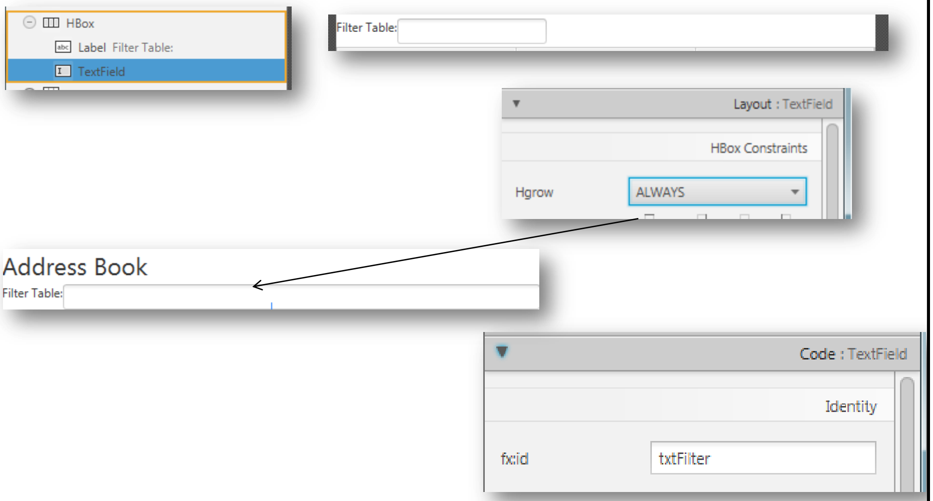
Address Book

Filter Table:

HoGent

Scene Builder

TextField



HBox
- Label Filter Table:
- TextField

Filter Table:

Layout : TextField

HBox Constraints

Hgrow: ALWAYS

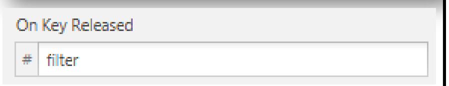
Address Book

Filter Table:

Code : TextField

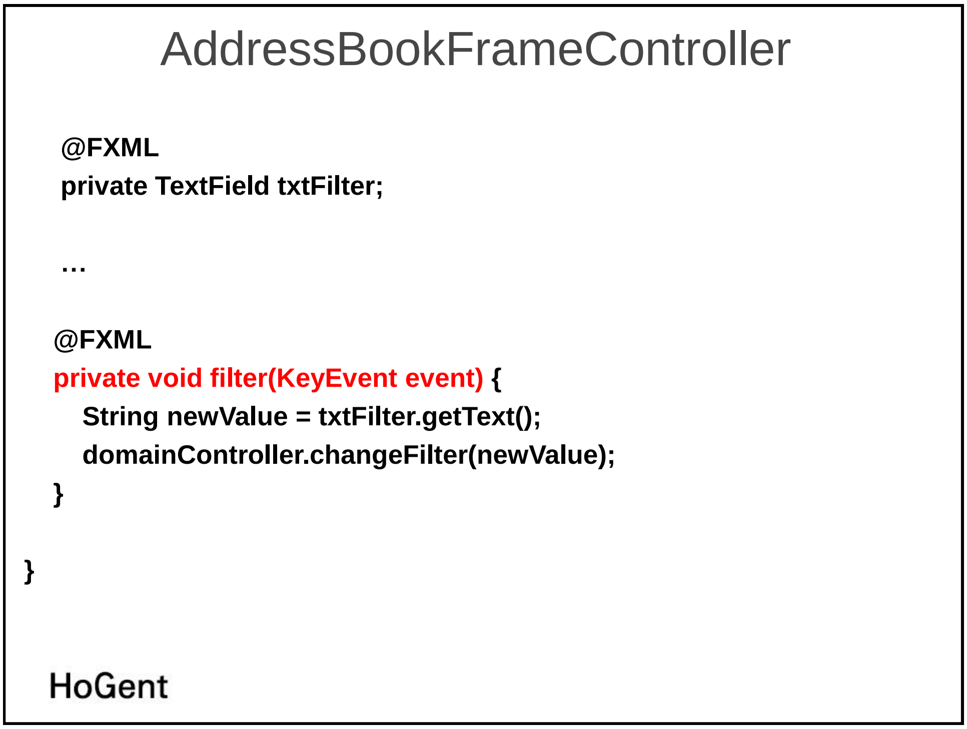
Identity

fx:id: txtFilter



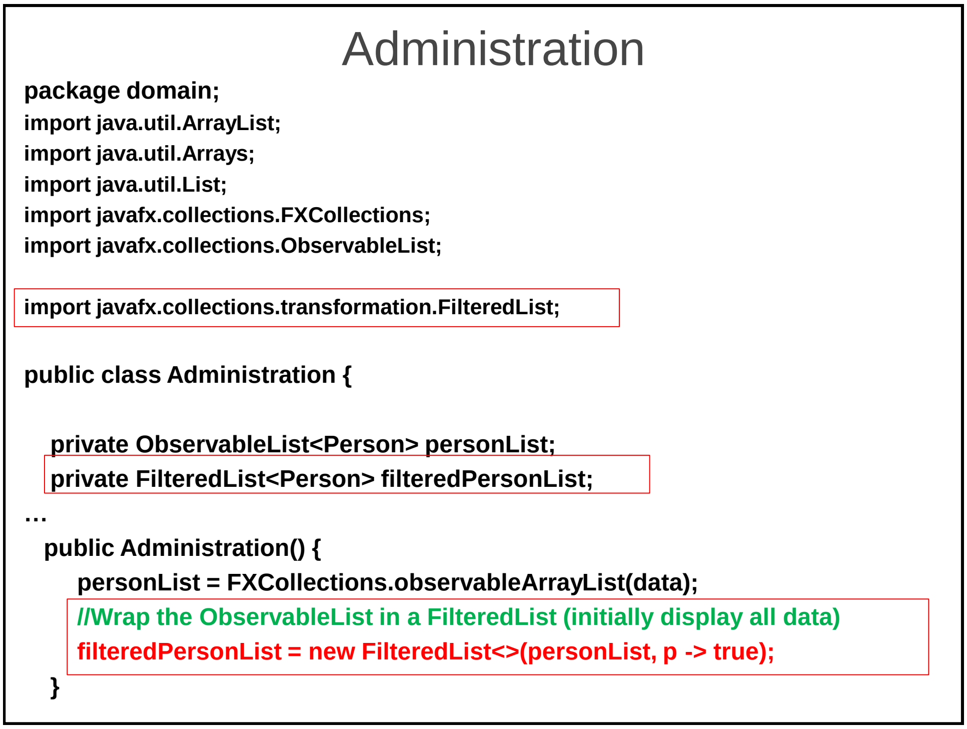
On Key Released: # filter

HoGent



AddressBookFrameController

```
  @FXML
  private TextField txtFilter;

  …

 @FXML
 private void filter(KeyEvent event) {
     String newValue = txtFilter.getText();
     domainController.changeFilter(newValue);
 }

}
```



Administration

```
package domain;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

import javafx.collections.transformation.FilteredList;

public class Administration {

    private ObservableList<Person> personList;
    private FilteredList<Person> filteredPersonList;
…
   public Administration() {
        personList = FXCollections.observableArrayList(data);
        //Wrap the ObservableList in a FilteredList (initially display all data)
        filteredPersonList = new FilteredList<>(personList, p -> true);
    }
```

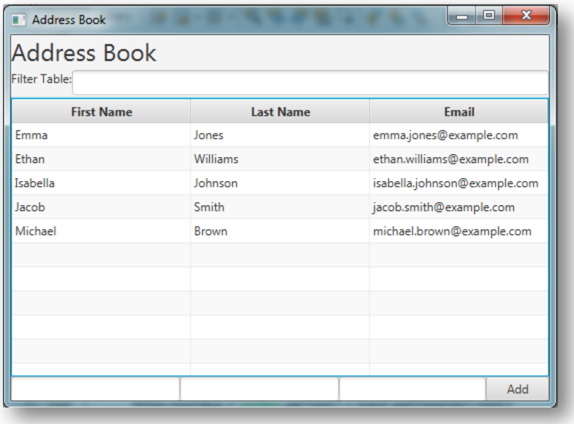
Administration

```
public ObservableList<Person> getPersonList() {
    //return FXCollections.unmodifiableObservableList(personList);
    return filteredPersonList;
}

public void editFirstName(int index, String newFirstName) {

        index = filteredPersonList.getSourceIndex(index);
        personList.get(index).setFirstName(newFirstName);
}

…
```

Administration

```
public void changeFilter(String filterValue) {
      filteredPersonList.setPredicate(person -> {
          // If filter text is empty, display all persons.
          if (filterValue == null || filterValue.isEmpty()) {
              return true;
          }
          // Compare first name and last name of every person with
          //filter text.
          String lowerCaseValue = filterValue.toLowerCase();
          return person.getFirstName().toLowerCase().contains(lowerCaseValue)
              || person.getLastName().toLowerCase().contains(lowerCaseValue);
      }
      );
}
```

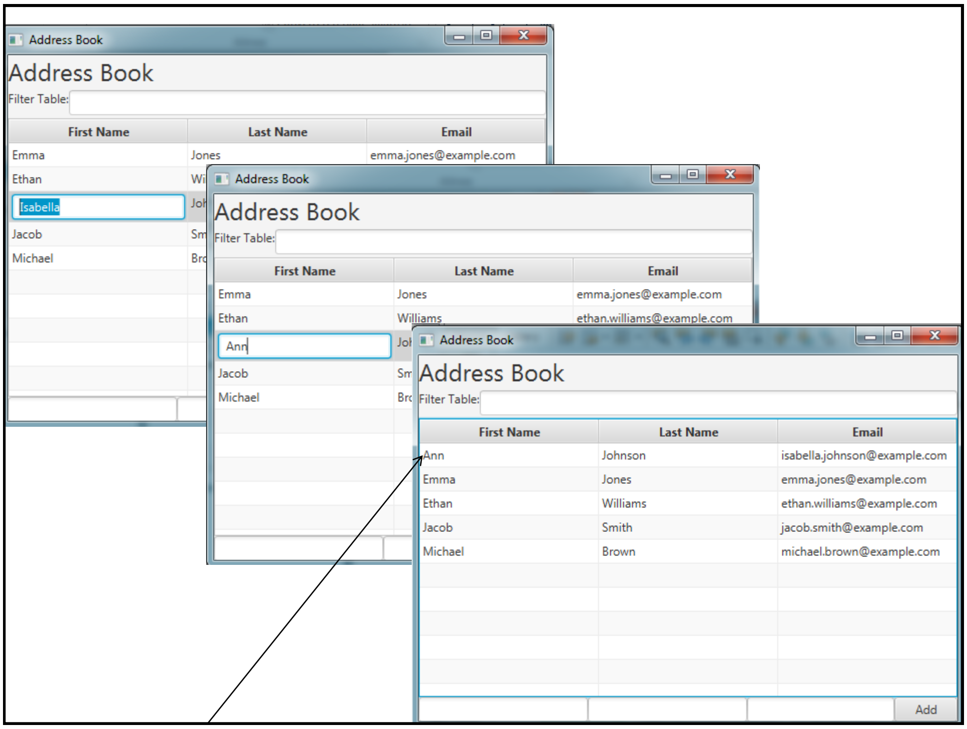
4.5 View sorteren (volgens voornaam, familienaam, email)



Address Book

Filter Table:

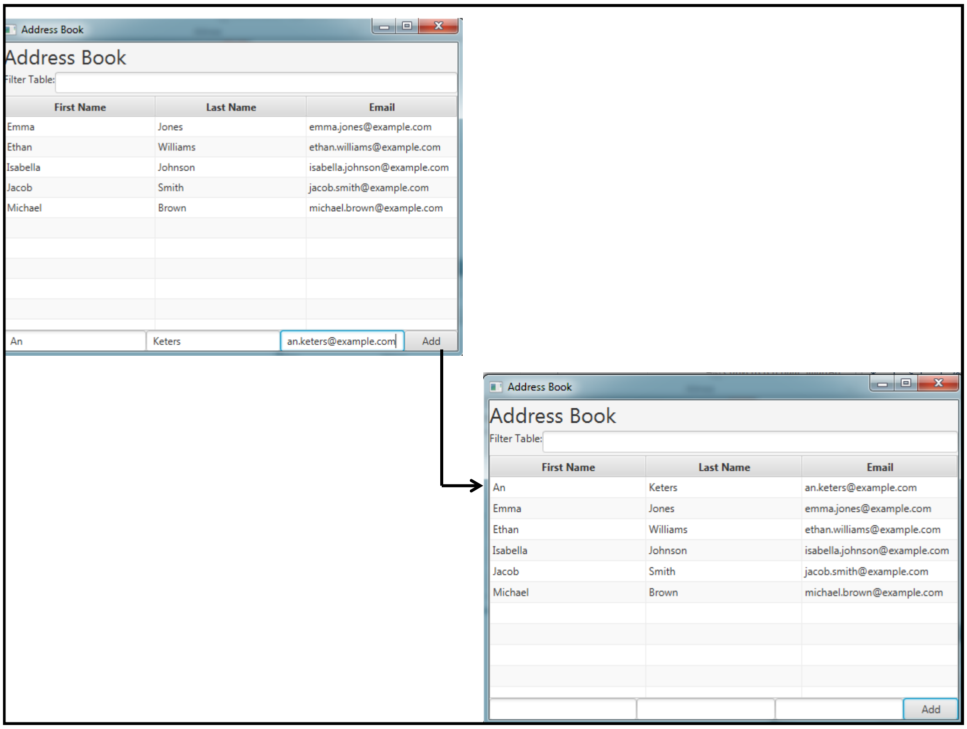
First Name	Last Name	Email
Emma	Jones	emma.jones@example.com
Ethan	Williams	ethan.williams@example.com
Isabella	Johnson	isabella.johnson@example.com
Jacob	Smith	jacob.smith@example.com
Michael	Brown	michael.brown@example.com



Address Book

Filter Table:

First Name	Last Name	Email
Ann	Johnson	isabella.johnson@example.com
Emma	Jones	emma.jones@example.com
Ethan	Williams	ethan.williams@example.com
Jacob	Smith	jacob.smith@example.com
Michael	Brown	michael.brown@example.com



Address Book

Filter Table:

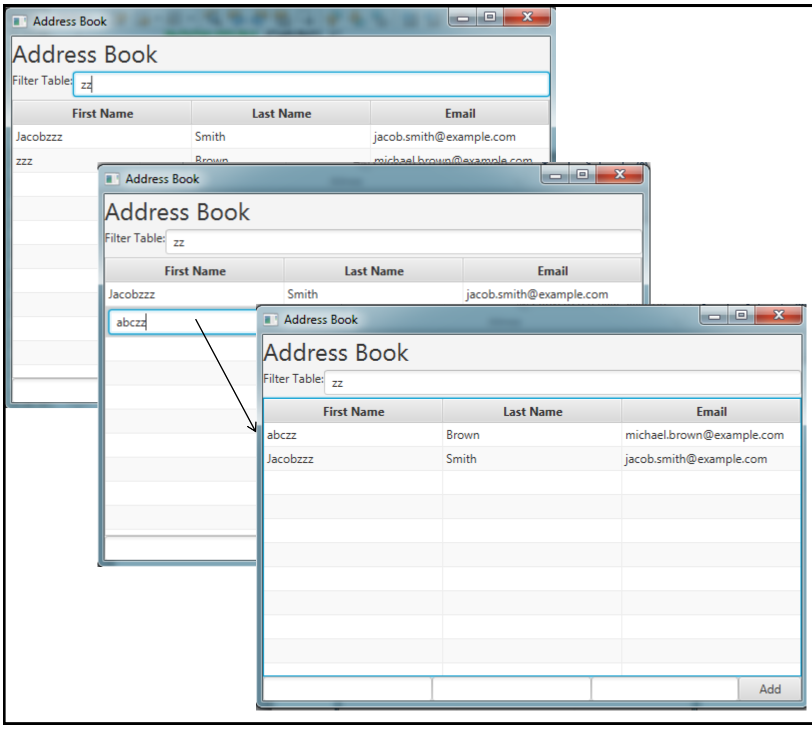
First Name	Last Name	Email
Emma	Jones	emma.jones@example.com
Ethan	Williams	ethan.williams@example.com
Isabella	Johnson	isabella.johnson@example.com
Jacob	Smith	jacob.smith@example.com
Michael	Brown	michael.brown@example.com

An Keters an.keters@example.com Add

Address Book

Filter Table:

First Name	Last Name	Email
An	Keters	an.keters@example.com
Emma	Jones	emma.jones@example.com
Ethan	Williams	ethan.williams@example.com
Isabella	Johnson	isabella.johnson@example.com
Jacob	Smith	jacob.smith@example.com
Michael	Brown	michael.brown@example.com



Address Book

Filter Table: zz

First Name	Last Name	Email
Jacobzzz	Smith	jacob.smith@example.com
zzz	Brown	michael.brown@example.com

Address Book

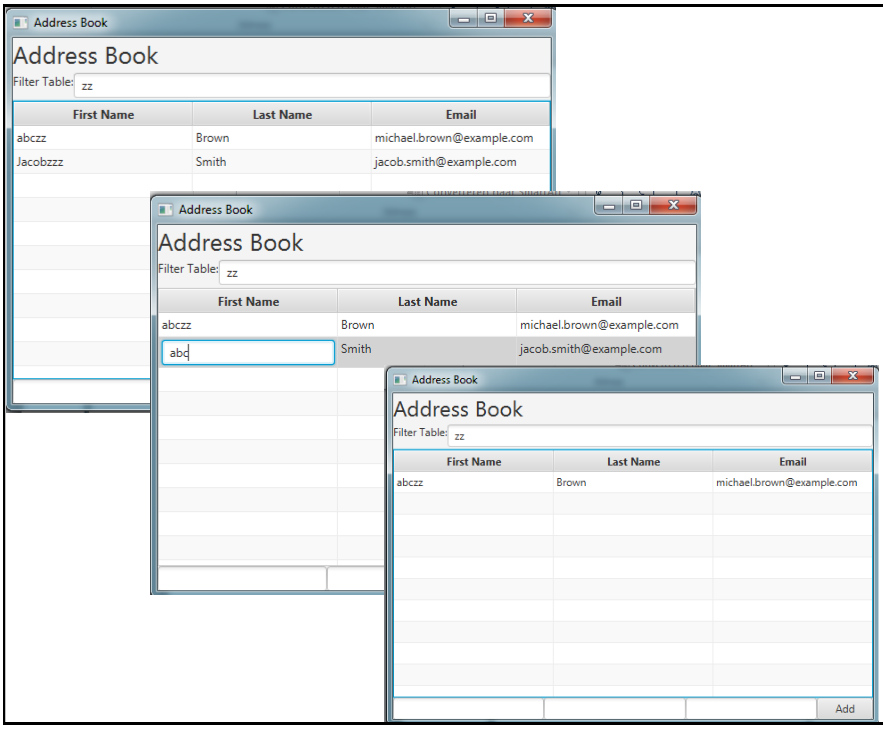
Filter Table: zz

First Name	Last Name	Email
Jacobzzz	Smith	jacob.smith@example.com
abczz		

Address Book

Filter Table: zz

First Name	Last Name	Email
abczz	Brown	michael.brown@example.com
Jacobzzz	Smith	jacob.smith@example.com



Address Book

Filter Table: zz

First Name	Last Name	Email
abczz	Brown	michael.brown@example.com
Jacobzzz	Smith	jacob.smith@example.com

Address Book

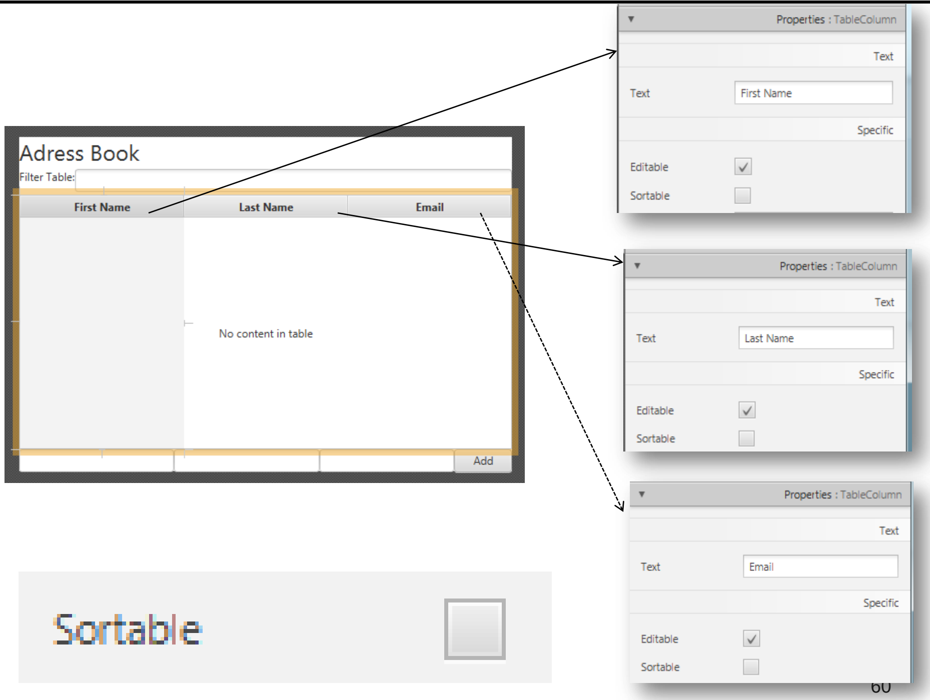
Filter Table: zz

First Name	Last Name	Email
abczz	Brown	michael.brown@example.com
abc	Smith	jacob.smith@example.com

Address Book

Filter Table: zz

First Name	Last Name	Email
abczz	Brown	michael.brown@example.com



Adress Book

Filter Table:

First Name	Last Name	Email

No content in table

Properties : TableColumn

Text

Text: First Name

Specific

Editable ☑

Sortable □

Properties : TableColumn

Text

Text: Last Name

Specific

Editable ☑

Sortable □

Properties : TableColumn

Text

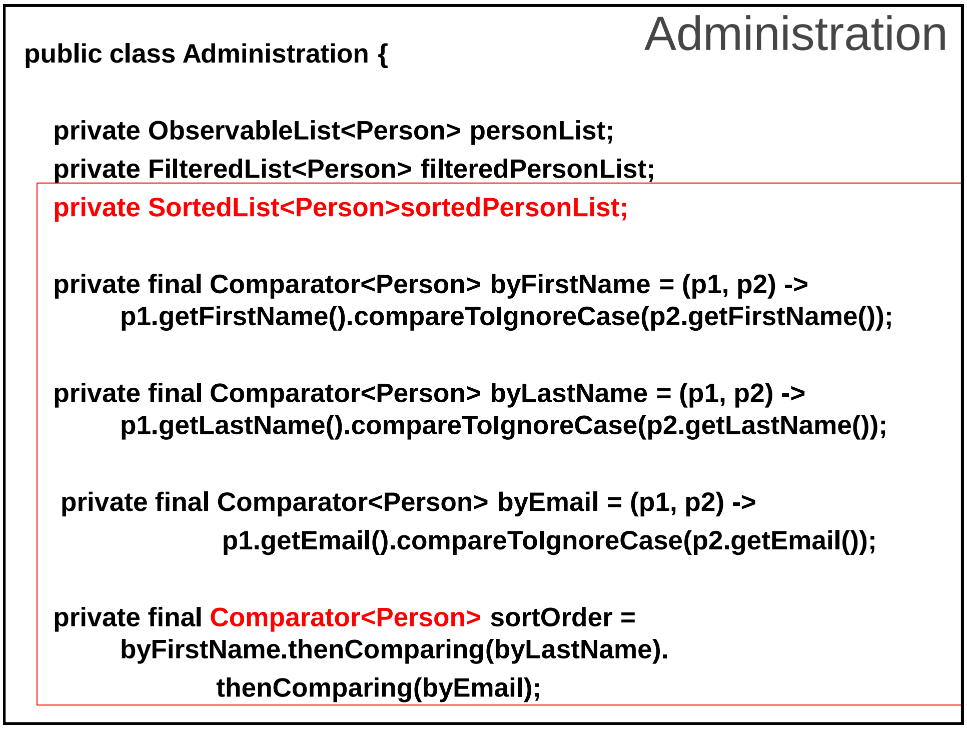
Text: Email

Specific

Editable ☑

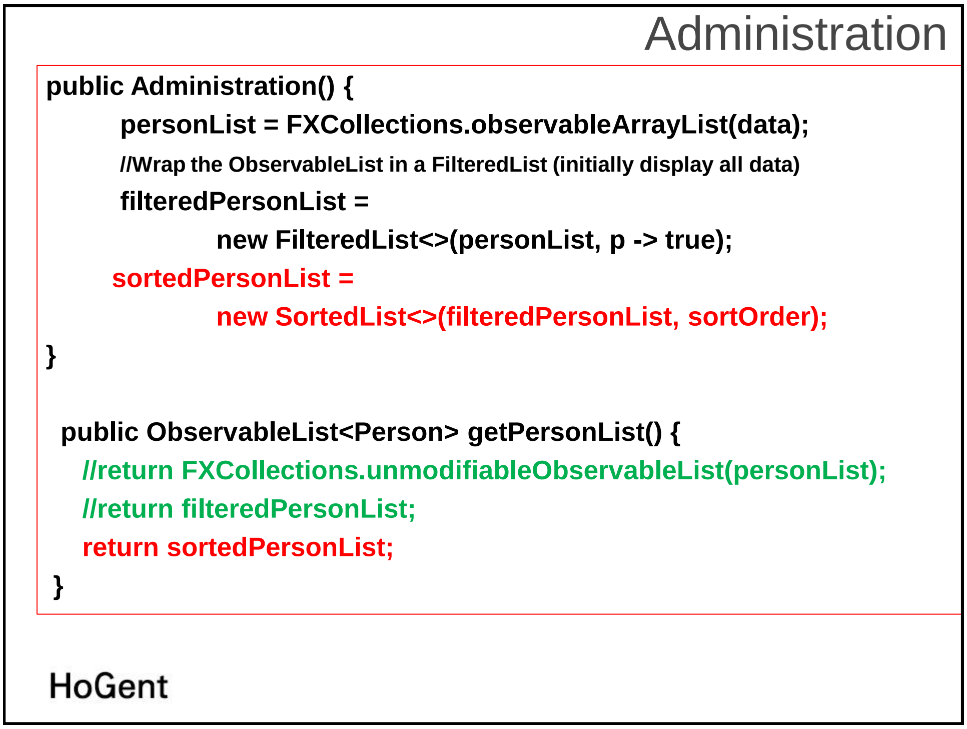
Sortable □

Sortable □



Administration

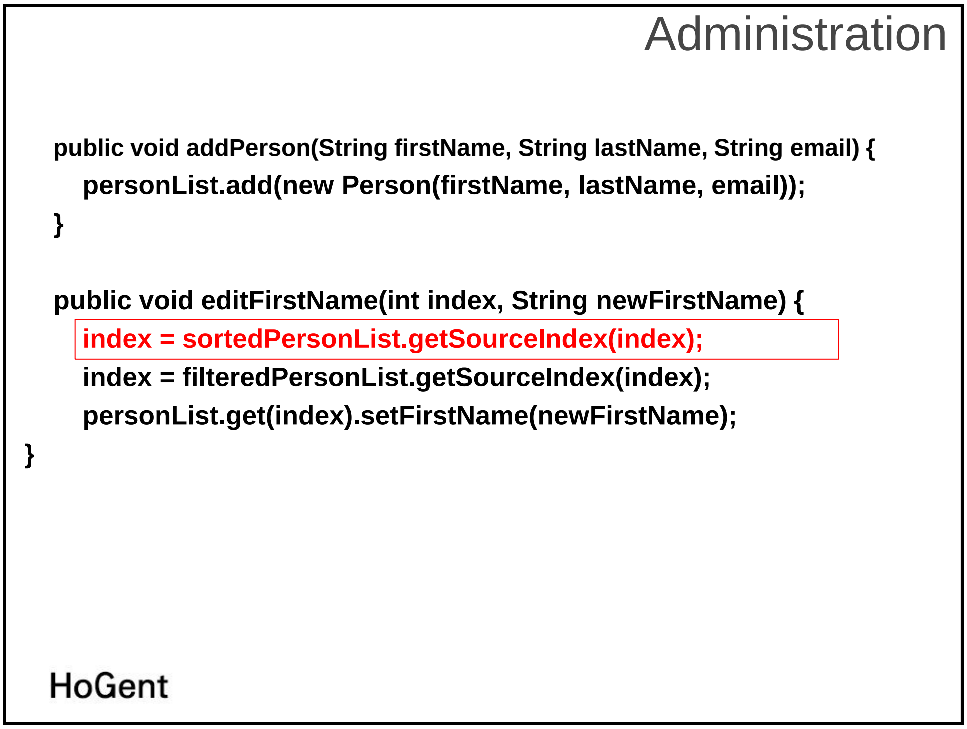
```
public class Administration {

    private ObservableList<Person> personList;
    private FilteredList<Person> filteredPersonList;
    private SortedList<Person>sortedPersonList;

    private final Comparator<Person> byFirstName = (p1, p2) ->
            p1.getFirstName().compareToIgnoreCase(p2.getFirstName());

    private final Comparator<Person> byLastName = (p1, p2) ->
            p1.getLastName().compareToIgnoreCase(p2.getLastName());

     private final Comparator<Person> byEmail = (p1, p2) ->
                      p1.getEmail().compareToIgnoreCase(p2.getEmail());

    private final Comparator<Person> sortOrder =
            byFirstName.thenComparing(byLastName).
                    thenComparing(byEmail);
```



Administration

```
public Administration() {
        personList = FXCollections.observableArrayList(data);
        //Wrap the ObservableList in a FilteredList (initially display all data)
        filteredPersonList =
                new FilteredList<>(personList, p -> true);
       sortedPersonList =
                new SortedList<>(filteredPersonList, sortOrder);
}

 public ObservableList<Person> getPersonList() {
    //return FXCollections.unmodifiableObservableList(personList);
    //return filteredPersonList;
    return sortedPersonList;
 }
```

HoGent



Administration

```
  public void addPerson(String firstName, String lastName, String email) {
    personList.add(new Person(firstName, lastName, email));
  }

  public void editFirstName(int index, String newFirstName) {
    index = sortedPersonList.getSourceIndex(index);
    index = filteredPersonList.getSourceIndex(index);
    personList.get(index).setFirstName(newFirstName);
}
```